ISP-Friendly Peer Matching without ISP Collaboration

Mohamed Hefeeda
(Joint work with ChengHsin Hsu)

9 December 2008
Introduction

- P2P Systems (file sharing, live streaming, VoD, ...) are very popular nowadays
  - Generating a major fraction of Internet traffic
- In most of these systems, peers offer limited capacity and reliability
- Client peer typically served by multiple senders
- Choosing “good” senders is important for
  - Clients: better performance
  - P2P System: more capacity, better scalability
  - Network (ISPs): reduced load on inter- /intra-ISP links
Our Work: Peer Matching Problem

- Given set of potential senders for a receiver, find subset of them that minimize the load on ISPs
  - Load = traffic on inter-ISP links (expensive) and on intra-ISP links

- General formulation/solution can be used in different Systems
  - BitTorrent-like: use in tracker
  - Gnutella-like: use in peers
  - Peer-assisted CDN: use in managing servers
Our Approach

- **Find nearby peers in terms of**
  - AS hops ➔ reduce inter-ISPs load
  - PoP (Point of Presence) distance ➔ reduce intra-ISPs load

- **Efficiently estimate AS paths & PoP distances**
  - Small memory/CPU requirements
  - Use public data (BGP tables, GeoIP, …), no probing

- **No infrastructure need to be deployed**
- **No need for cooperation from ISPs**
- **No need to modify client software**
Related Work

- **Provider Portal for P2P (P4P) [Xie 08]**
  - ISPs deploy servers (iTrackers) to guide peer matching
  - Require cooperation from ISPs and infrastructure

- **Using DNS redirections [Choffnes 08]**
  - Peers observe similar delay to CDN replicas close to each other
  - Require modifying clients

- **Match peers within the same AS [Bindal 06]**
  - Specific to BitTorrent
  - Peers outside AS are chosen randomly
Overview

- Large ISP: PoPs at many locations
- S3 is better for R than S2, though same AS distance
Overview

- We need to estimate AS hops between peers
  - Need AS path inference algorithm (valley-free policy)
  - Modified Dijkstra Algorithm [Mao 05] → very expensive, took ~2 days of running time, huge memory footprint
  - Our optimization: runs 1-3 hours, < 1% memory

- We also need to infer PoP topology
  - Propose simple method to cluster IP prefixes
  - Use GeoIP databases to map clusters to PoPs
Peer Matching

- **Distance oracle**: concise data structure, enables online matching in $O(1)$ steps
AS Path Inference

- **Original algorithm [Mao 05]**
  - $O(V^3)$, $V$ number of ASes
  - Currently, there are 28,000+ ASes [CAIDA]
  - Very expensive

- **Observation**
  - Many ASes have small degree (client/stub ASes)
  - 83% of ASes have 1 or very few C2P links [CAIDA]
  - their AS paths can be inferred using their providers’ paths
  - stub ASes can be removed from the AS graph
AS Path Inference

- **T**: transit AS,  **S**: stub AS
- **Given** T₀ – T₁ & T₀—T₂ ➔ T₀—S can be computed

- **We can recursively remove ASes**
  - Stub AS first
  - Transit ASes that are customers of larger ASes, and so on
### AS Path Inference

- **Estimate AS paths for “core” ASes first**
  - Using any algorithm
- **Then paths starting/end at a transit AS**
- **Then, the rest**
- **Time Complexity:** \( O(T^3 + TS + S^2) \)
  - \( T \) small fraction (<18%) of ASes \( \Rightarrow \) much faster
  - 3 hours (instead of 2 days) for whole matrix

<table>
<thead>
<tr>
<th>Source AS</th>
<th>Destination AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T Transit ASes</td>
<td>S Stub ASes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mohamed Hefeeda
Distance Oracle

- Maintaining the whole distance matrix is costly
  - Assume each entry takes 1 byte
  - #of ASes = 28,594 \(\Rightarrow\) 817 MB for distance matrix

- Idea: Maintain only the “core” matrix, other entries can be computed online

- Tradeoff: small core vs. more online computations
## Distance Oracle: Tradeoff

<table>
<thead>
<tr>
<th>Level</th>
<th>Core Size (#ASes)</th>
<th>Memory (MB)</th>
<th>Online Computations (Average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28,594</td>
<td>817</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>5,144</td>
<td>26</td>
<td>2.6</td>
</tr>
<tr>
<td>2</td>
<td>2,949</td>
<td>8</td>
<td>12.4</td>
</tr>
<tr>
<td>3</td>
<td>2,471</td>
<td>6</td>
<td>97.0</td>
</tr>
</tbody>
</table>

- We can use a distance oracle of size 8 MB, and perform < 13 operations online on average.  
  - #operations depends on AS degree, not size of AS graph.
PoP Distance: Overview

- Infer approximate PoP topology of ASes

Algorithm overview:
- Collect all IP prefixes from BGP tables and map them to their ASes
- For each AS, cluster IP prefixes into one or more PoPs
- Analyze BGP updates to infer connectivity among PoP in same AS
- Combine PoP topology with AS graph
Evaluation

- Trace-based simulations
- Use real IPs from
  - BitTorrent tracker: ~150,000
  - Online content provider (CBC -- Canadian Broadcasting Corporation); ~160,000
- BGP tables from RouteViews & RIPE
- AS relationship data from CAIDA
- Use GeoIP [maxmind.com] for IP/AS/location mapping
Evaluation

- Simulate multi-sender sessions for objects with different popularities: 1% -- 10%
- Change number of chosen senders: 10 – 100
- Implement
  - ISPF: ISP-Friendly Matching
  - ISPF-Lite: ISP-Friendly Matching, without PoP info
  - AS: Match based on AS distance only
  - Prefix: Match based on longest common IP prefix
    - Uses IPs of peers \( \Rightarrow \) almost free
  - Random: currently used in many systems

Mohamed Hefeeda
Sample Results: Vary #Senders

- Some gain from (the free) Prefix
- Significant gain from ISPF/ISPF-Lite
Sample Results: Vary Object Popularity

- Even more gain for rare objects
Sample Results: PoP Distance

- ISPF achieves shorter distances
Dynamics of AS Graph

- CAIDA AS relationship data for a full year
- Run AS inference algorithm once a month or so
Conclusions and Future Work

- Peer matching is important in multi-sender systems
  - P2P systems, distributed streaming, CDN, pCDN, ...

- Peer matching algorithms based on
  - AS distance
  - PoP distance

- Do not need infrastructure, modifying peers, ISP cooperation

- Efficient computation and storage of AS distance
  - ~ 8 MB (<1% of whole matrix) for 28,000+ ASes
Conclusions and Future Work

- Currently implementing in pCDN (for CBC) and in BitTorrent tracker

- Exploring link cost model
  - Assign weights to inter-AS links

- Exploring more efficient methods for discovering & storing PoP topology
Thank You!

Questions??

- More info at:

http://nsl.cs.sfu.ca/